*Decomposition Design Pattern*

*Decomposition patterns, which enables splitting the application into a set of loosely coupled services. There are three primary decomposition approaches.*

* *Decomposition by Business Capability - This approach is based on business capability decomposition, which aims to create microservices aligned with the business needs and can be developed and maintained independently.* For example, a banking application might have business capabilities for account management, transaction processing, and loan origination. Each of these capabilities could be supported by a separate microservice.

1. *Business Alignment: By organizing microservices around business capabilities, aligning software development with business goals and objectives becomes easier.*
2. *Cross-Functional Teams: This pattern encourages cross-functional teams that are focused on specific business capabilities.*
3. *Improved User Experience: By focusing on specific business capabilities, microservices can be optimized for the needs of end-users.*

*Issues –*

1. *Difficulty in identifying business capabilities.*
2. *Tight coupling between microservices*
3. *Complexity in coordinating microservices*

* *Decomposition by Transaction - In this pattern, the components that participate in a transaction are grouped together as part of the same microservice. This grouping helps to avoid problems with two-phase commit and reduces latency issues that may arise in a distributed system.*

*Benefits –*

1. *Reduced complexity*
2. *Minimized two-phase commit problems.*
3. *Reduced latency.*
4. *Better alignment with business needs.*

*Issues –*

1. *Potential for monolith creation.*
2. *Increased cost and complexity*
3. *Incompatible versions*

* *Decomposition by Subdomain - This subdomain is a process of breaking down a monolithic system into smaller, independent microservices based on the corresponding subdomains defined by Domain-Driven Design (DDD)*
* *Ex. Products – Buy and Sale, Marketing – media, reports*

*Advantages –*

1. *Improved scalability and predictability.*
2. *Loosely coupled architecture.*